Subject: Re: Dynamically resizing arrays
Posted by Andrew[2] on Sat, 05 Feb 2005 04:04:08 GMT
View Forum Message <> Reply to Message

Hi Jonathan,

Assume you have declared your initial array, lets call it array_init

array_init=INTARR(100) ;for arguments sake

if we now assume that you are in the loop and want to append the new
data, which we also assume is 100 columns long (i.e fixed length)

```
new_row=intarr(100) ;the array of new data
FOR i=0,99 DO BEGIN
  ;some operation or whatever you do here
  ;to place the data in new_row

  array_init=[[array_init],[new_row]] ;appened it to the original data
ENDFOR
```

your array_init will now grow one row at a time with each loop. You
might want to consider using a WHILE statement though to avoid the FOR
loop. I hope this helps, and is correct. Try it with some dummy arrays
(I did).

Cheers
Andrew

Jonathan Greenberg wrote:
> I was hoping to get some feedback on the best way of creating a
"database"
> -- an array of fixed columns but unknown number of rows which will be
> appended to within some sort of loop. What is the best way of doing
this?
>
> --j

Subject: Re: Dynamically resizing arrays
Posted by netnews.comcast.net on Sun, 06 Feb 2005 04:08:28 GMT
View Forum Message <> Reply to Message

Adding data row by row like this can be a little inefficient as you are
forcing IDL to allocate RAM for every iteration. I have found it to be
a bit quicker to grow the array in chunks appropriate for your application.

Taking Andrew's example:

```
array_init=INTARR(100,100) ; allocate 100 rows initially
nrows = 0
totrows = 100

while (have_new_data = 1) do begin
   array_init[*,nrows] = your_row_of_data
   ++nrows

   ;  grow array if needed - adding 100 rows at a time
   if (nrows eq totrows - 1) then begin
      array_init = [[array_init],[INTARR(100,100)]]
      totrows = totrows + 100
    endif

endwhile

;  trim array when done adding data
array_init = array_init[*,nrows - 1]
```

You have a little more to keep track of but for larger arrays it will be
worth the hassle.

-Rick

Andrew wrote:

```
>  Hi Jonathan,
>
>  Assume you have declared your initial array, lets call it array_init
>
>  array_init=INTARR(100) ;for arguments sake
>
>  if we now assume that you are in the loop and want to append the new
>  data, which we also assume is 100 columns long (i.e fixed length)
>
>  new_row=intarr(100) ;the array of new data
>  FOR i=0,99 DO BEGIN
>    ;some operation or whatever you do here
>    ;to place the data in new_row
>
>    array_init=[[array_init],[new_row]] ;appened it to the original data
>  ENDFOR
>
>  your array_init will now grow one row at a time with each loop. You
>  might want to consider using a WHILE statement though to avoid the FOR
```

> loop. I hope this helps, and is correct. Try it with some dummy arrays
> (I did).
>
> Cheers
> Andrew
>
> Jonathan Greenberg wrote:
>
>> I was hoping to get some feedback on the best way of creating a
>
> "database"
>
>> -- an array of fixed columns but unknown number of rows which will be
>> appended to within some sort of loop. What is the best way of doing
>
> this?
>
>> --j
>
>

Subject: Re: Dynamically resizing arrays
Posted by Michael Wallace on Mon, 07 Feb 2005 17:01:11 GMT
View Forum Message <> Reply to Message

Rick Towler wrote:
> Adding data row by row like this can be a little inefficient as you are
> forcing IDL to allocate RAM for every iteration. I have found it to be
> a bit quicker to grow the array in chunks appropriate for your application.

[snip]

> ; grow array if needed - adding 100 rows at a time
> if (nrows eq totrows - 1) then begin
> array_init = [[array_init],[INTARR(100,100)]]
> totrows = totrows + 100
> endif

One other common solution is to multiply your current number of rows by 2 rather than adding a constant. If your array grows past the initial mark you've set, this exponential algorithm will require fewer array resize operations than the linear algorithm, and hence improve efficiency. The one caveat is that if your array grows really large, doubling the array might use up way too much memory. There are benefits from both approaches -- it just depends on the nature of your data and how large you think it may grow.

-Mike

Subject: Re: Dynamically resizing arrays
Posted by kenklare on Wed, 09 Feb 2005 19:00:38 GMT
View Forum Message <> Reply to Message

Adding a row at a time requires time and memory to grow as the square
of the size.
Memory grows because the new chunk cannot reuse the old one, it is too
big.
Adding a chunk at a time reduces the coefficient by 100 or whatever.
Doubling increases time logarithmically.
This square growth can noticibly slow a program that does a lot of it.